```
case 3:
printf("Third\n");
break;

default:
printf("Wrong Choice\n");
}
```

→ Function : Self contained subprogram meant to perform well defined specific task

→ Avoid repetition of code

→ Functions can be stored in library.

FUNCTIONS

↙ Built-in function

↓

Predefined and are present in C Library

Ex: sqrt(), printf(), scanf(), etc.

↘ User defined function

↓

1. Function declaration or prototype
2. Function call
3. Function definition

```
# include <stdio.h>
void drawline(void);    → Function declaration
int main()
{
    drawline();    → Function call
}
void drawline(void);
{
    int i;
for (i=1; i<=80; i++)
  {
    printf("-");
  }
}
```

} Function definition

Based on return type and argument ; we categorise function in 4 categories :

1. Function with no arguments and no return value
2. Function with no arguments and a return value
3. Function with arguments and no return value
4. Function with arguments and a return value.

```
# include <stdio.h>
int      sum(int, int);
main()
{
    int a,b,s;
    printf("Enter the value for a and b: ");
    scanf("%d %d", &a, &b);

    s = sum(a,b)
    printf("Sum of %d and %d is \n", a,b);
    }

    int  sum(int x, int y)
    {  int s;
       s = x+y;
       return s;
       }
```

→ WAP to find sum of digits of any number

```
 # include <stdio.h>
  int   sum(int, int);
  main ()
   {
      int num;
      printf ("Enter the no: ");
      scanf ( "%d", &num);
      printf ("sum of digits of %d is %d", num, sum(num));
   }
```

```
int sum(int)
{
  int i, sum=0, rem;
   while (n>0)
  { rem= n%10;
     sum+= rem;
     n=n/10;
   }
     return (sum);
  }
```

→ Recursion

The function that calls itself (inside function body) again and again is known as recursive function.

```
main()                         rec()
{ ...                          { ...
  rec();                        if (...) /* Terminating
}                               condition*/
                                 ....
                                 rec();
                                }
```

$$n! = \begin{cases} 1 & , n=0 \\ n*(n-1)! & , n>0 \end{cases}$$

$f(n) = n * f(n-1)$
$f(n-1) = (n-1) * f(n-2)$

```
# include <stdio.h>
 int fact(int)
 main ()
{
   int num;
   printf("Enter a no: ");
scanf ("%d", &num);
printf ("Factorial of %d is %d", num, fact(num));
}

int  fact(~~of fed~~ int n)
{
     if  (n==0)
         return (1);

   else
         return (n* fact(n-1));
}
```

$$a^n = \begin{cases} 1, & n=0 \\ a * a^{n-1}, & n>0 \end{cases}$$

```
# include <stdio.h>
 float power (float a, int n);
  main ()
  {
     float  a, p;
     int  n;
     printf ("Enter a and n:   ");
     scanf ("%f %d", &a, &n);
      p=power (a,n);
      printf ("%f raised to the power of %d is %f \n", a,
                n, p);
      float power (float a, int n)
   }

   { if  (n==0)
         return (1);
   else
       return  (a*power(a,n+1));
     }
```

Fibonacci Series

$$\begin{tikzcd}
0 \arrow[r, bend right] & 1 \arrow[r, bend right] & 1 \arrow[r, bend right] & 2 \arrow[r, bend right] & 3 \arrow[r, bend right] & 5 \arrow[r, bend right] & 8
\end{tikzcd}$$

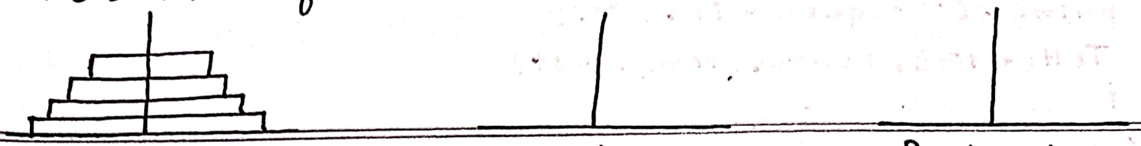
$$f(n) = \begin{cases} 1 & , \quad n=0 \text{ or } n=1 \\ f(n-1)+f(n-2) & , \quad n>0 \end{cases}$$

```
#   include <stdio.h>
    int  fib(int);
    main()
  {
      int   nterms, i ;
      printf ("Enter no.of terms:   ");
      scanf ("%d", &nterms);

       for (i=0;  i<nterms, i++)
        printf("%d", fib(i));
        printf ("\n");
    }

      int fib (int n)
    {
       if  (n==0 || n==1)
             return 1;
      else
      return   (fib (n-1)+ fib(n-2));
    }
```

Tower of Hanoi



Source Tower   Temperory   Destination

→ bigger disk cannot be placed on smaller plate.

→ One disk can move at a time.

ToH (n, source, temp, dest) =

Move disk from S to D , n=1

ToH (n, S, D, T) , n>1
Move nth disk. from S→D :

ToH (n-1, T, S, D)
Move (n-1)th disk from T→D

```
#include <stdio.h>
 ToH ( int ndisk, char source, char temp, char dest)
  {
        if  (ndisk >0)
      {  ToH (ndisk-1, source, dest, temp);
         printf ("Move disk %d %c → %c \n", ndisk, source,
                 dest");
         ToH (ndisk-1, temp, source, dest);
        }
     }

   main ()
 {
   char source = 'S',
   Temp= 'T', Dest = 'D';
    int ndisk;
  Printf ("Enter the no. of disk");
```